Improved Parallel Algorithm for Time Series Based Forecasting Using OTIS-Mesh

Ashish Gupta
Department of Computer Science and Engineering
Indian School of Mines , Dhanbad, Jharkhand, 826004, India
Email: ashish.parj@gmail.com

***Abstract*-Forecasting always plays an important role in business, technology and many others and it helps organizations to increase profits, reduce lost sales and more efficient production planning. A parallel algorithm for forecasting reported recently on OTIS-Mesh[9]. This parallel algorithm requires $5(\sqrt{n}-1)$ electronic steps and 4 optical steps. In this paper we present an improved parallel algorithm for time series short term forecasting using OTIS-Mesh. This parallel algorithm requires 5(-1) electronic steps and 1 optical step using same number of I/O ports as considered in [9] and shown to be an improvement over the parallel algorithm for time series forecasting using OTIS-Mesh [9].**

***Index Terms*- Time Series Forecasting, Parallel Algorithm, OTIS-Mesh.**

I. INTRODUCTION

Many businesses, Organizations, get benefits through forecasting in terms of profit increment, reduce lost sales and also it helps to make production planning more efficient. Forecasting plays an important role in many areas such as weather forecasting, flood forecasting etc. Many researchers implemented forecasting on many different interconnection networks in parallel. Forecasting can also be map on OTIS network. Optical Transpose Interconnection System (OTIS) [1],[2] is basically a hybrid architecture which is benefits from optical and electronic connections. Optical connection is used to connect the processors when the distance between the processors exceeds the few millimetres (in other package) and electronic connections are used to connect the close processors (within the same physical package). Several models exploit the idea of optical and electronic connections. In an OTIS-Mesh, n^2 processors are divided into n groups where processors in each group follow × 2D mesh layout. According to the OTIS rule , G^{th} group is connected to the P^{th} processor and P^{th} group is connected to the G^{th} processor. The Pattern of OTIS can be varied according to the interconnection among the processor in the group. The topology of OTIS-Mesh shown in figure 1.

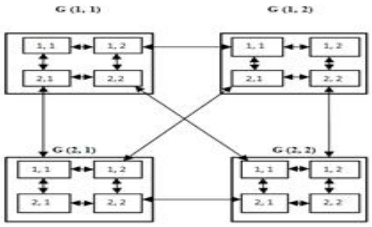


Figure 1. OTIS-Mesh network

II. RELATED WORKS

Many Researchers has developed parallel algorithms for short term time series forecasting. Jana and sinha presented two parallel algorithms for forecasting implemented on linear array and complete binary tree model[14] and require

m+1steps and $(m-n+2)+\log_2 n$ steps respectively and in extended case[14] it requires $\frac{n}{p}(m-n+1)+p-1$ and $\frac{n}{p}(m-n+2)+\log_2 p$ steps respectively on ST array and ST tree. Both the algorithm based on weighted moving average techniques. Sudhanshu and jana presented parallel algorithm for forecasting based on OTIS-Mesh Network[9]. This parallel algorithm requires $5(\sqrt{n}-1)$ electronic moves and 4 optical moves[9]. In this paper we present an improved parallel algorithm for forecasting based on OTIS- Mesh network. This parallel based on weighted moving average technique and requires $5(\sqrt{n}-1)$ electronic moves and 1 optical move. This parallel algorithm can be compared to parallel algorithm for forecasting as considered in [9].

III. FORECASTING MODELS

Forecasting models can be divided in to Qualitative and Quantitative forecasting models. We discuss here the time series models of quantitative forecasting model. Among different quantitative forecasting models available for successful implementation of decision making systems, time series models are very popular. In these models, given a set of past observations, say $d_1,d_2,\ldots,d_m$, the problem is to estimate d(m + t) through extrapolation, where t(called the lead time) is a small positive integer and usually set to 1. The observed data values usually show different patterns, such as constant process, cyclical and linear trend as shown in Figure 2. Several models are available for time series forecasting. However, a particular model may be effective for a specific pattern of the data, e.g. simple moving average is very suitable when the data exhibits a constant process. Weighted moving average is a well known time series model for short term forecasting which is suitable when the data exhibits a cyclical pattern around a constant trend. Exponential weighted moving average is more widely accepted technique method for short term forecasting than the (simple) weighted moving average. However, our motivation to parallelize weighted moving average with the fact that both the exponential weighted moving average and the simple moving average (MA) are the special cases of the weighted moving

DOI: 01.IJIT.01.03.519

average as will be discussed in section IV. Moreover, in order to find the optimum value of the window size, it involves O(m) iterations where each iteration requires O(n) time for calculating (m – n + 1) weighted moving averages for a window size n and m data size. In this paper, we present an improved parallel algorithm for short term forecasting which is based on weighted moving average of time series model and mapped on OTIS-Mesh. This algorithm is shown to be an improvement over the algorithm as considered in [9] and requires $5(\sqrt{n}$"1) electronic moves + 1 OTIS move using same number of I/O ports for m size data set and n window size using n^2 processors.

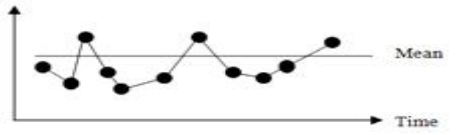


Figure 2(a). Constant data pattern

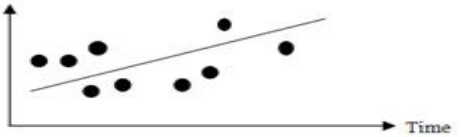


Figure 2(b). Trend data pattern

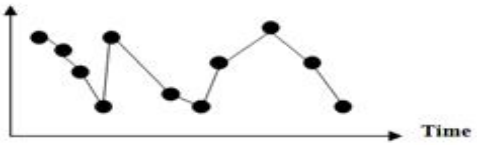


Figure 2(c). Cyclic data pattern

The rest of the paper is organized as follows. Section IV describes the methodology for time series forecasting using weighted moving average with its special cases. In section V, we present our proposed parallel algorithm. We also discuss scalability issue for forecasting.

IV. MEHODOLOGY

We describe here the methodology for forecasting using weighted moving average as given in [14]. In this method, for a set of n data values d_t, d_{t+1},…, d_{t-n+1} and a set of positive weights w_1, w_2, …, w_n, we calculate their weighted moving average at time t by the following formula

$$W^M(t) = \frac{w_n d_t + w_{n-1} d_{t-1} + w_{n-2} d_{t-2} + \ldots + w_1 d_{t-n+1}}{w_n + w_{n-1} + w_{n-2} + \ldots + w_1}. \quad (1)$$

Where $w_n w_{n-1} \geq w_{n-2} \geq, \ldots, \geq w_1 \geq 0$. We then use $W^M(t)$ to estimate the forecast value $\hat{d}(t+\tau)$ at time $t+\tau$, i.e., $(t+\tau) = W^M(t)$. The quality of the forecast depends on the selection of the window size (n). Therefore, in order to find the optimum value of n, we calculate m – n + 1 weighted averages for a specific value of n by sliding the window over the data values and the corresponding mean square error (MSE) is also calculated using

$$MSE = \sum_{t=n+\tau}^{m} \frac{[d_t - \hat{d}_t]^2}{m-n-\tau+1}. \quad (2)$$

We then vary the window size (n) to obtain the corresponding MSE with the newly calculated weighted moving averages. The same process is repeated for n = 1, 2, 3,..,m. The value of n for which MSE is least is chosen for forecasting.

Special Cases

A.Simple Moving Average:

In this method, equal importance is given to each data value. Hence we assign equal weight 1/n to all the data values and obtain the following moving average.

$$\partial^M(t) = \frac{d_t + d_{t-1} + d_{t-2} + \ldots + d_{t-n+1}}{n}. \quad (3)$$

As we have discussed in section III, this method is best when the data pattern shows a constant process.

B. Exponential Moving Average:

In this method, the more recent observations are given a larger weight to face smaller error and thus the weights are assigned in decreasing order. The formula for exponential moving average is as follows

$$E^M(t) = \alpha d_t + \alpha(1-\alpha) d_{t-1} + \alpha(1-\alpha)^2 d_{t-2} + \ldots + \alpha(1-\alpha)^n d_{t-n+1}. \quad (4)$$

where weight $w_i = \alpha(1-\alpha)^{n-i}$ $1 \leq i \leq n$, $1 \leq \alpha \leq n$. This method is suitable for a cyclical pattern around a constant trend and is widely accepted specially for business environment. However, the method suffers from the proper selection of the value of the parameter and there is no easy method to do it.

V. ALGORITHM

Assume $\tau = 1$. Then (m – n + 1) weighted moving averages are obtained from equation (1) for a given window size n along with their error term as follows

$$\left.\begin{aligned} W^M(n) &= \frac{w_1 d_1 + w_2 d_2 + w_3 d_3 + \ldots + w_n d_n}{w_1 + w_2 + w_3 + \ldots + w_n} \\ W^M(n+1) &= \frac{w_1 d_2 + w_2 d_3 + w_3 d_4 + \ldots + w_n d_{n+1}}{w_1 + w_2 + w_3 + \ldots + w_n} \\ W^M(n+2) &= \frac{w_1 d_3 + w_2 d_4 + w_3 d_5 + \ldots + w_n d_{n+2}}{w_1 + w_2 + w_3 + \ldots + w_n} \\ &\vdots \\ W^M(m) &= \frac{w_1 d_{m-n+1} + w_2 d_{m-n+2} + w_3 d_{m-n+3} + \ldots + w_n d_m}{w_1 + w_2 + w_3 + \ldots + w_n} \end{aligned}\right\} \quad (5)$$

$$\left.\begin{aligned} E(n+1) &= d_{n+1} - W^M(n) \\ E(n+2) &= d_{n+2} - W^M(n+1) \\ E(n+3) &= d_{n+3} - W^M(n+2) \\ &\vdots \\ E(m) &= d_m - W^M(m-1) \end{aligned}\right\} \quad (6)$$

DOI: 01.IJIT.01.03.519

$$\text{MSE} = \sum_{i=n+1}^{m} \frac{[E_i^2]}{m-n}. \qquad (7)$$

For a different value of n say n_i, $1 \leq i \leq m$, we require to compute different set of $(m - n_i + 1)$ weighted moving averages (as given above) for a maximum of m iterations. However, our target is to parallelize the above computation for a single iteration so that the overall time complexity can be significantly reduced. The basic idea is as follows. We initially feed the data values and the weight vector through the boundary processors. Then using suitable electronic and OTIS moves, they are stored in the D and W registers respectively. Next we calculate their products for each processor in parallel. The products are then used to form the local sum in each group which are finally accumulated using suitable electronic and OTIS moves to produce weighted moving averages.

VI. PARALLEL ALGORITHM

Step 1. */*Data Input*/*

1.1Feed the data values d_i's, $1 \leq i \leq m$ to the boundary processors in the 1st column position of each group G_{xy}, $1 \leq x,y \leq \sqrt{n}$ as shown in Figure 3.

1.2 Feed the weights w_j's, $1 \leq j \leq n$ to the boundary processors in the 1st processor P_{11} of each group G_{xy}, $1 \leq x,y \leq \sqrt{n}$ as shown in figure 3.

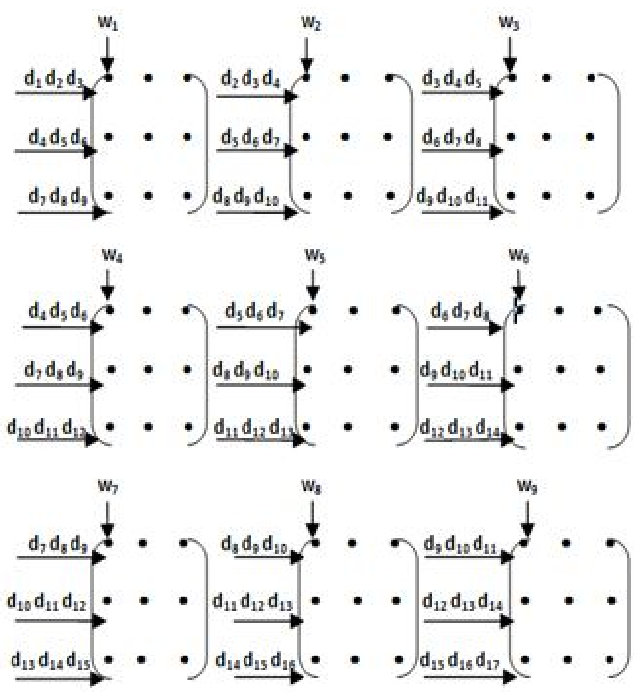


Figure 3. Feeding of data and weights

Step 2. */* Data distribution into D-registers */*
Shift the data values row-wise to store them in D-registers in a pipeline fashion.

Step 3. */* Broadcast of weights */*
Perform column-wise broadcast on the weights fed in step 1.2 in parallel and store them in W register.

Step 4. */*Broadcast of weights*/*
Perform row-wise broadcast on the weights stored in step 3 of each group in parallel and store them in W register.
Illustration 1: Content of W registers after step 4 shown in figure 5.

W_1	W_1	W_1	W_2	W_2	W_2	W_3	W_3	W_3
W_1	W_1	W_1	W_2	W_2	W_2	W_3	W_3	W_3
W_1	W_1	W_1	W_2	W_2	W_2	W_3	W_3	W_3
W_4	W_4	W_4	W_5	W_5	W_5	W_6	W_6	W_6
W_4	W_4	W_4	W_5	W_5	W_5	W_6	W_6	W_6
W_4	W_4	W_4	W_5	W_5	W_5	W_6	W_6	W_6
W_7	W_7	W_7	W_8	W_8	W_8	W_9	W_9	W_9
W_7	W_7	W_7	W_8	W_8	W_8	W_9	W_9	W_9
W_7	W_7	W_7	W_8	W_8	W_8	W_9	W_9	W_9

Figure 5. Content of W registers after step 4

Step 5. $\forall$ Processors do in parallel
Perform OTIS move on the content of W registers in each group.
Illustration 2: Content of W registers after step 5 shown in figure 6.

W_1	W_2	W_3	W_1	W_2	W_3	W_1	W_2	W_3
W_4	W_5	W_6	W_4	W_5	W_6	W_4	W_5	W_6
W_7	W_8	W_9	W_7	W_8	W_9	W_7	W_8	W_9
W_1	W_2	W_3	W_1	W_2	W_3	W_1	W_2	W_3
W_4	W_5	W_6	W_4	W_5	W_6	W_4	W_5	W_6
W_7	W_8	W_9	W_7	W_8	W_9	W_7	W_8	W_9
W_1	W_2	W_3	W_1	W_2	W_3	W_1	W_2	W_3
W_4	W_5	W_6	W_4	W_5	W_6	W_4	W_5	W_6
W_7	W_8	W_9	W_7	W_8	W_9	W_7	W_8	W_9

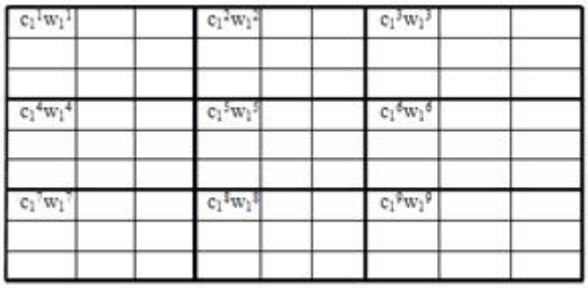
Figure 6. Content of W registers after step 5

Step 6. $\forall$ Processors do in parallel
Form the products with the contents of D registers and W registers and store it in C-register.

Step 7. */*Perform Summation*/*
$\forall$ groups do steps 7.1 and 7.2 in parallel
7.1 Sum up with the content of C register column wise and store it in first row processors in each group.
7.2 Sum up with the content of W registers column wise and store it in first row processors in each group.

*Step 8. /*Parallel summation row wise*/*
$\forall$ groups do steps 8.1 and 8.2 in parallel
8.1 Sum up with the content of C registers in first row of each group and store it in C registers of first processor P_{11} in each group.
8.2 Sum up with the content of W registers in first row of each group and store it in first processor P_{11} in each group.
Illustration 3: We store the data , weights, and products in D_i^j, W_i^j and C_i^j registers $1 \leq i,j \leq n$ respectively where i indicates register number in group and j indicates group number. Group numbers are organized in row major order. C registers and W registers of processor P_{11} after step 8 shown in figure 7.

DOI: 01.IJIT.01.03.519



Step 9. Divide the content of C register and W register and store it in R registers of first processor of each group $G_{xy,}$ $1\leq x,y\leq\sqrt{n}$.

Remark 1: The final results emerge from the R- register of first processor P_{11} in each Group (G_{xy},P_{11}) $1\leq x,y\leq\sqrt{n}$ The result after step 8 shown in table I.

RESULTS

We describe here the time complexity required to map the parallel algorithm on OTIS-Mesh on n^2 processors.

Time Complexity: Steps 2, 3, 4, 7, 8 requires $(\sqrt{n}\text{-}1)$ steps each. Step 5 requires 1 OTIS move. Rest of the steps requires constant time. Therefore the parallel algorithm requires $5(\sqrt{n}\text{-}1)$ electronic moves and 1 OTIS move.

Scalability: Now we consider any arbitrary size of the window to map the above algorithm on a $\sqrt{n}\times\sqrt{n}$ OTIS-mesh. In other words, we consider the case when the window size is independent of the number of processors. For the sake of simplicity and without any loss of generality, let us assume it to be kn. Note that in this case, the size of the data set will be 2kn " 1. Accordingly the data set is also partition into k subsets: $\{d_1, d_2,\ldots, d_n\}$, $\{d_2, d_3,\ldots,d_{n+1}\}$, $\ldots,\{d_{2kn-n}, d_{2kn-n+1}, \ldots, d_{2kn-1}\}$. Given a subset of the data, its corresponding weights, then we can partition the weight set into k subsets: $\{w_1, w_2, \ldots, w_n\}$, $\{w_{n+1}, w_2,\ldots, w_{2n}\}\ldots$, $\{w_{(k-1)n+1}, w_{(k-1)n+2}, \ldots, w_{kn}\}$.weight subset is fed to the $\sqrt{n}\times$ OTIS-mesh. We then run the above algorithm (Parallel Algorithm) and store the result temporarily. Next we input another data subset along with the corresponding weight subset, execute Parallel Algorithm and update the current result with the previously calculated partial result.

This process is repeated k times to yield the final result. This is obvious to note that this version of the algorithm requires 5k("1) electronic moves + 1k OTIS moves, which is k times more than time complexity of Parallel Algorithm.

TABLEI.
CONTENT OF C REGISTERS AND W REGISTERS OF PROCESSORS P11 OF EACH GROUP AFTER STEP 8

$C_1^1 = d_1w_1 + d_2w_2 + d_3w_3 + d_4w_4 + d_5w_5 + d_6w_6 + d_7w_7 + d_8w_8 + d_9w_9$ $W1^1 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$	$C_1^2 = d_2w_1 + d_3w_2 + d_4w_3 + d_5w_4 + d_6w_5 + d_7w_6 + d_8w_7 + d_9w_8 + d_{10}w_9$ $W1^2 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$	$C_1^3 = d_3w_1 + d_4w_2 + d_5w_3 + d_6w_4 + d_7w_5 + d_8w_6 + d_9w_7 + d_{10}w_8 + d_{11}w_9$ $W1^3 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$
$C_1^4 = d_4w_1 + d_5w_2 + d_6w_3 + d_7w_4 + d_8w_5 + d_9w_6 + d_{10}w_7 + d_{11}w_8 + d_{12}w_9$ $W1^4 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$	$C_1^5 = d_5w_1 + d_6w_2 + d_7w_3 + d_8w_4 + d_9w_5 + d_{10}w_6 + d_{11}w_7 + d_{12}w_8 + d_{13}w_9$ $W1^5 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$	$C_1^6 = d_6w_1 + d_7w_2 + d_8w_3 + d_9w_4 + d_{10}w_5 + d_{11}w_6 + d_{12}w_7 + d_{13}w_8 + d_{14}w_9$ $W1^6 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$
$C_1^7 = d_7w_1 + d_8w_2 + d_9w_3 + d_{10}w_4 + d_{11}w_5 + d_{12}w_6 + d_{13}w_7 + d_{14}w_8 + d_{15}w_9$ $W1^7 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$	$C_1^8 = d_8w_1 + d_9w_2 + d_{10}w_3 + d_{11}w_4 + d_{12}w_5 + d_{13}w_6 + d_{14}w_7 + d_{15}w_8 + d_{16}w_9$ $W1^8 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$	$C_1^9 = d_9w_1 + d_{10}w_2 + d_{11}w_3 + d_{12}w_4 + d_{13}w_5 + d_{14}w_6 + d_{15}w_7 + d_{16}w_8 + d_{17}w_9$ $W1^9 = w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9$

window size is independent of the number of processors. For the sake of simplicity and without any loss of generality, let us assume it to be kn. Note that in this case, the size of the data set will be 2kn -- 1. Accordingly the data set is also partition into k subsets: $\{d_1, d_2,\ldots, d_n\}$, $\{d_2, d_3,\ldots,d_{n+1}\}$, $\ldots,\{d_{2kn-n}, d_{2kn-n+1}, \ldots, d_{2kn-1}\}$. Given a subset of the data, its corresponding weights, then we can partition the weight set into k subsets: $\{w_1, w_2, \ldots, w_n\}$, $\{w_{n+1}, w_2,\ldots, w_{2n}\}\ldots$, $\{w_{(k-1)n+1}, w_{(k-1)n+2}, \ldots, w_{kn}\}$.weight subset is fed to the $\sqrt{n}\times\sqrt{n}$ OTIS-mesh. We then run the above algorithm (Parallel Algorithm) and store the result temporarily. Next we input another data subset along with the corresponding weight subset, execute Parallel Algorithm and update the current result with the previously calculated partial result.

This process is repeated k times to yield the final result. This is obvious to note that this version of the algorithm requires $5k(\sqrt{n}-1)$ electronic moves + 1k OTIS moves, which is k times more than time complexity of Parallel Algorithm.

CONCLUSIONS

In this paper, we have presented an improved parallel algorithm for short term forecasting using weighted moving average technique. The algorithm is mapped on n^2 processor OTIS-Mesh. We have shown that it requires $5(\sqrt{n}-1)$ electronic moves + 1 OTIS move. This Parallel algorithm shown to be an improvement over the Parallel algorithm using same number of I/O ports as considered in [9].The algorithm

DOI: 01.IJIT.01.03.519

is also shown to be scalable.

COMPARATIVE RESULT ANALYSIS

In Table II we compare the time complexity in terms of electronic moves and optical moves required by the parallel algorithms mapped on OTIS-Mesh network. In our proposed algorithm, we require same time complexity in terms of electronic moves as compared to [9] but we require only 1 optical move.

TABLE II.
COMPARISON OF OTIS-MESH BASED PARALLEL ALGORITHMS

Parallel Algorithm	Electronic Moves	Optical Moves
Sudhanshu and Jana[9	$5(\sqrt{n}-1)$	4
Proposed Algorithm	$5(\sqrt{n}-1)$	1

FUTURE WORKS

For implementation of forecasting in parallel architecture, proper shifting of data should be done properly. So we should try to explore the parallel architecture for proper shifting of data. We can also exploit the properties of OTIS based networks. Therefore we should try to map time series forecasting on other OTIS based networks such as OTIS-Mesh Of Trees, OTIS-Hypercube etc.

REFERENCES

[1]. G.C Marsden, P.J Marchand, P. Harvey and S, C, Esener, "Optical transpose interconnection system architecture," Optical Letters, Vol 18, No. 13,pp 1083-1085,july,1993.
[2]. C. F. Wang and S. Sahani, "OTIS optoelectronic computers" Parallel Computation using Optical Interconnection,"K Li, Y Pan and S.Q Zhang, Eds.Khuwer Academic 1988.
[3]. Wang C. F. and Sahni S.,"Image processing on the OTIS-Mesh optoelectronic Computer," IEEE Trans. On Parallel and Distributed Systems, 11, 97-109, 2000.
[4]. Wang C. F. and Sahni S.,. "Matrix multiplication on the OTIS-Mesh optoelectronic computer," IEEE Transactions on Computers, 50(July 2001), 635 – 646, 2001.
[5]. Wang C. F. and Sahni S, "Basic operations on the OTIS-Mesh optoelectronic computer," IEEE Trans. on Parallel and Distributed Systems 9(Dec. 1998) 1226–1998, 1998.
[6]. Wang C. F. and Sahni S, "BPC Permutations on the OTIS-Hypercube, optoelectronic computer," Informatica,22(3) 1998.
[7]. Jana P. K. and Sinha B. P., "An Improved Parallel Prefix Algorithm on OTIS-Mesh," Parallel Processing Letters, 2006, 16, 429-440.
[8]. Jana P. K, "Polynomial interpolation and polynomial root finding on OTIS-Mesh," Parallel Computing, 32(4), 301-312, 2006.
[9]. Sudhanshu Kumar Jha, Prasanta K. Jana, "Parallel Algorithm for Timeseries Based Forecasting using OTIS-Mesh," international journal of Computer Application(0975-8887) 2010, Volume 1 – No. 26.
[10]. Lucas K. T. and Jana P. K., "An efficient parallel sorting algorithm on OTIS Mesh of Trees," Proc. IEEE Intl. Advance Computing Conference , (6-7 March, 2009), Patiala, India, 175-180 ,2009.
[11]. Lucas K. T., Mallick D. K. and Jana P. K.,. "Parallel Algorithm for conflict graph on OTIS triangular array," Lecture Notes in Computer Science, 4904, 274-279, 2008.
[12]. Rajasekaran S. and Sahni S, "Randomized routing selection, and sorting on the OTIS-Mesh," IEEE Transaction on Parallel and Distributed Systems, 9, 833-840, 1998.
[13]. Wheelwright S. C., and Makridakis S.,"Forecasting methods for management," 1980 ,John Wiley and Sons.
[14]. Jana P. K., Sinha B. P., "Fast Parallel Algorithms for Forecasting," Computers Math. Applic. 34(9) 39-49, 1997.
[15]. Nassimi, D., and Sahni, S.,. "Bitonic sort on a Mesh connected parallel computer," IEEE Trans. Comput. C-28(1) ,1979.

DOI: 01.IJIT.01.03.519